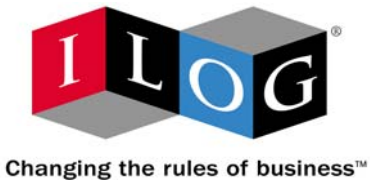

ILOG CPLEX 9.0 Release Notes

October 2003

Copyright © 1987-2003, ILOG, S. A. — All rights reserved.

Table of Contents

Conversion Notes for All Users 6
Error Codes Removed 6
New Settings for Existing Parameters 6
Simplex Feasibility Tolerance 7
Lower Bound on Converted Discrete Variables 7
New Default of Basis Interval Parameter 8
Barrier Variable Upper Bound Parameter Removed 8
Simplex Threads Parameter Removed 8
Deprecated QP File Format 8
Conversion Notes for Microsoft Users 9
Conversion Notes for Concert Technology Users 9
Rounding of Integer Variables 9
IloIntVar, IloIntVarArray and IloNumVar, IloNumVarArray 10
IloRange: Changes in Expressions 10
Method isQuadratic Replaced by isQO or isQC. 11
Conversion Notes for Interactive Optimizer Users 11
New Features: a Checklist. 12
New Parameters 13
New Settings for Familiar Parameters 13
New Methods in Concert Technology 14

New Routines in the Callable Library 14
New Commands in the Interactive Optimizer 15
Improved and Additional Error Messages 15
New Features: More Detail 16
Quadratic Constraints 17
QP Barrier Crossover 19
New Setting for MIP Emphasis 19
Relaxation Induced Neighborhood Search (RINS) 20
Integrality Tolerance 20
Finding IIS after Dual Simplex 21
Feasible Optimization 21
Concert Technology for C#.NET Users 21
Parallel Optimizers in Java 22
Automatic Translation of Logical Constraints 22
XML and Concert 23
IloXmlContext 23
IloXmlInfo 23
IloXmlReader 23
IloXmlWriter 23
CSV File Format 24
IloCsvReader 24
IloCsvLine 24
IloCsvReader::Iterator 24
New Settings for Advanced Basis Indicator 25
Preprocessing and Factorization (Uncrush) 26
New Settings for Preprocessing Dependency Indicator 27
New Devex Setting for Dual Pricing Indicator 28
New Setting for MIP Strategy: Dive Type 29
New Default Value for Barrier Growth Parameter 29
Improved Documentation 30

Index 31

ILOG CPLEX 9.0 Release Notes

These release notes highlight improvements and new features in ILOG CPLEX 9.0. Please review these notes before using ILOG CPLEX 9.0.

- *Conversion Notes for All Users* on page 6
- *Conversion Notes for Microsoft Users* on page 9
- *Conversion Notes for Concert Technology Users* on page 9
- *Conversion Notes for Interactive Optimizer Users* on page 11
- *New Features: a Checklist* on page 12
- *New Features: More Detail* on page 16

Conversion Notes for All Users

For users of versions 8.0 and 8.1 of ILOG CPLEX, the following topics offer guidelines for easy migration to ILOG CPLEX version 9.0. (Users of versions prior to 8.0 must first apply the Conversion Notes accompanying ILOG CPLEX 8.0 and 8.1 before upgrading to ILOG CPLEX 9.0.)

For a complete list of machine types and library formats (including version numbers of compilers and JDKs) see the file *yourCPLEXhome*`/mptable.html`.

Conversion notes possibly of interest to all users include these topics:

- *Error Codes Removed* on page 6
- *New Settings for Existing Parameters* on page 6
- *Simplex Feasibility Tolerance* on page 7
- *Lower Bound on Converted Discrete Variables* on page 7
- *New Default of Basis Interval Parameter* on page 8
- *Barrier Variable Upper Bound Parameter Removed* on page 8
- *Simplex Threads Parameter Removed* on page 8
- *Deprecated QP File Format* on page 8

Error Codes Removed

These error codes no longer exist:

`CPXERR_DBL_MIN`

`CPXERR_BOUNDS_BINARY`

`CPXERR_IIS_NO_PRIMAL`

The conditions that could raise those error codes no longer occur.

New Settings for Existing Parameters

Several new features are controlled by new values of already existing parameters. The existence of these values should not affect current applications, but for completeness the following list indicates where a change has been made. Consult *New Features: More Detail* on page 16 of this document for details about what the new settings do. Two of these parameters (`AdvInd` and `DepInd`) entail special considerations in the Interactive Optimizer;

see *Conversion Notes for Interactive Optimizer Users* on page 11 for more information about them.

Table 1 *New Settings for Existing Parameters*

Parameter in Concert Technology	Parameter in Callable Library	Parameter in Interactive Optimizer	New Settings
AdvInd	CPX_PARAM_ADVIND	advance	0 (default), 1, 2
DepInd	CPX_PARAM_DEPIND	preprocessing dependency	-1, 0, 1, 2, 3
DiveType	CPX_PARAM_DIVETYPE	mip strategy dive	3
DPriInd	CPX_PARAM_DPRIIND	simplex dgradient	5 CPX_DPRIIND_DEVEX
MIPEmphasis	CPX_PARAM_MIPEMPHASIS	mip emphasis	4 CPX_MIPEMPHASIS_HIDDENFEAS
NodeAlg	CPX_PARAM_SUBALG	mip strategy subalgorithm 0	0 (default) CPX_ALG_AUTOMATIC
BarGrowth	CPX_PARAM_BARGROWTH	barrier limits growth	Default: 1e12

Simplex Feasibility Tolerance

This note pertains to MIP models.

The ILOG CPLEX MIP optimizer no longer automatically resets simplex feasibility tolerance to be less than or equal to the integrality tolerance. This new feature may have implications for users who relied on the automatic reset in previous releases. For more detail, see *Integrality Tolerance* on page 20.

Lower Bound on Converted Discrete Variables

ILOG CPLEX will now automatically set the lower bound to 0.0 (zero) when you change a semi-continuous variable or a semi-integer variable to any one of these types:

- continuous,
- integer,
- binary.

The impact of this change is that the resulting model will be a true relaxation of the MIP model containing that variable; that is, any MIP feasible solution to the original model will also be feasible for the changed variant.

New Default of Basis Interval Parameter

In previous releases, the default value of the basis interval parameter was 50000. Consequently, ILOG CPLEX used to write basis files every fifty-thousand iterations. ILOG CPLEX no longer does that. Now the default value of that parameter is 2100000000, that is, the largest integer parameter value in ILOG CPLEX. If you want to set a different value, set the basis interval parameter:

- `BasInterval` in Concert Technology
- `CPX_PARAM_BASINTERVAL` in the Callable Library
- `set simplex basisinterval` in the Interactive Optimizer

Barrier Variable Upper Bound Parameter Removed

The parameter that controlled the upper limit on variables in the barrier optimizer has been removed. ILOG CPLEX now assumes an infinite upper bound unless a finite bound is actually stated in the model. That obsolete parameter was known as:

- `BarVarUp` in Concert Technology
- `CPX_PARAM_BARVARUP` in the Callable Library
- `set barrier limits varupper` in the Interactive Optimizer

Simplex Threads Parameter Removed

The parameter to limit the number of simplex threads has been removed. This obsolete parameter was known as:

- `SimThreads` in Concert Technology
- `CPX_PARAM_SIMTHREADS` in the Callable Library
- `set simplex limits threads` in the Interactive Optimizer

Deprecated QP File Format

In future releases, the QP file format will no longer be supported. In this release, QP file format is deprecated; that is, it still exists, but is not recommended for use.

Conversion Notes for Microsoft Users

Please note that Microsoft Corporation has announced plans to discontinue standard support for the Microsoft MSVC6 compiler as of September 30, 2003. Accordingly, ILOG will no longer be able to provide standard support for those ILOG customers that have not upgraded their applications to a later version of the Microsoft compiler. Nevertheless, ILOG will continue to provide bug fixes for these customers, to the extent ILOG can do so without the support of Microsoft.

Also please note that the standard distribution of ILOG CPLEX 9.0 contains these subdirectories or folders of interest to Microsoft users:

- *yourCPLEXhome*`\lib\msvc6\stat*`
- *yourCPLEXhome*`\lib\msvc7\stat*`
- *yourCPLEXhome*`\lib\x86_.net2003_7.1\stat*`
- *yourCPLEXhome*`\bin\win32`

where *yourCPLEXhome* indicates the place on your Microsoft platform where you installed the product, and `bin` and `lib` are distinct folders there.

Documentation for `CPLEX.NET` is available in compiled Microsoft Help format (CHM) in *yourCPLEXhome*`\doc\windows.`

Conversion Notes for Concert Technology Users

These changes in Concert Technology may affect existing ILOG CPLEX applications:

- *Rounding of Integer Variables* on page 9
- *IloIntVar, IloIntVarArray and IloNumVar, IloNumVarArray* on page 10
- *IloRange: Changes in Expressions* on page 10
- *Method isQuadratic Replaced by isQO or isQC* on page 11

Rounding of Integer Variables

A change in the rounding conventions of integer variables in Concert Technology may affect existing user applications.

When numeric bounds are assigned to an integer variable, they are inward rounded to an integer value in Concert Technology. Recall that bounds can be assigned to an integer variable in these ways:

- created by the constructor of `IloIntVar` with bounds at the time of creation;

- created by the constructor of `IloNumVar` with the type `ILOINT`;
- changed by the modifiers `setLB` and `setUB`.

These examples illustrate the effect of this rounding convention:

```
IloIntVar x(env, 2, 10);
x.setUB(6.5);
x.getUB() will return 6
x.setLB(2.2);
x.getLB() will return 3;
x.setUB(8);
x.getUB() will return 8;
```

IloIntVar, IloIntVarArray and IloNumVar, IloNumVarArray

A change in the inheritance of `IloIntVar` and `IloNumVar` in Concert Technology may affect existing user applications.

In earlier versions of Concert Technology, the class `IloIntVar` derived from `IloNumVar`. As a consequence of this inheritance, an array of numeric variables `IloNumVarArray` could contain instances of `IloIntVar` as elements. Likewise, methods that accepted `IloNumVar` or `IloNumVarArray` as arguments could also accept `IloIntVar` or `IloIntVarArray`.

However, in Concert Technology 2.0, the class `IloIntVar` no longer derives from `IloNumVar`. To accommodate this change in the class hierarchy, new methods have been added to Concert Technology and ILOG CPLEX to accept instances of `IloIntVar` and `IloIntVarArray` where arguments of type `IloNumVar` or `IloNumVarArray` only used to appear.

User applications that relied on the inheritance of `IloIntVar` from `IloNumVar` may have to be adjusted to this change in the class hierarchy.

IloRange: Changes in Expressions

For users of Concert Technology, there has been a change with respect to the class `IloRange`.

Concert 2.0 no longer creates `IloRange` objects for constraints of the form:

```
expression ~ expression // changed
```

where ~ is a relation. Now `IloRange` objects are created **only** for constraints of the form:

```
expression ~ value
```

and

```
value <= expression <= value
```

All interfaces involved in this change have been extended to work with `IloConstraint` instead. For examples of what has changed, see also the *Concert Technology Release Notes*.

Method isQuadratic Replaced by isQO or isQC

Because of the introduction of quadratically constrained programming (QCP) models in ILOG CPLEX 9.0, the method `cplex.isQuadratic` is deprecated. Instead, use the methods:

- `isQO` to detect the presence of a quadratic objective function;
- `isQC` to detect the presence of one or more quadratic constraints.

Conversion Notes for Interactive Optimizer Users

In order to support new capabilities, two parameters have changed in the Interactive Optimizer, from a `yes` or `no` setting, to a choice among integer values. These parameters appear in Table 2.

Table 2 *Syntax Change for Two Parameters in Interactive Optimizer*

Parameter	Old Default	New Default	Acceptable Settings
`advance`	`yes`	1	0,1,2
`preprocessing dependency`	`no`	-1	-1, 0, 1, 2, 3

For more information about the additional choices now available, see *New Settings for Advanced Basis Indicator* on page 25 and *New Settings for Preprocessing Dependency Indicator* on page 27.

The command `display problem stats` in the Interactive Optimizer now displays better organized and more detailed results. In particular, the information displayed now varies according to the setting of `read datacheck` (which may be `yes` or `no`). The command also now displays additional output to support quadratically constrained problems.

For example, in previous versions, you may have seen something like this:

```
Problem name: sc205.mps.gz
Constraints        :     205  [Less: 114,  Equal: 91]
Variables          :     203
Constraint nonzeros:     551
Objective  nonzeros:       1
RHS        nonzeros:      38
```

In this new version, with the default `datacheck` setting `no`, you see something like this:

```
(default datacheck setting]:
Problem name: sc205.mps.gz
Variables            :     203
Objective nonzeros   :       1
Linear constraints   :     205  [Less: 114,  Equal: 91]
  Nonzeros           :     551
  RHS nonzeros       :      38
```

In this new version, with the `read datacheck` setting `yes`, you see something like this:

```
(datacheck "yes"):
Problem name: sc205.mps.gz
Variables            :     203
Objective nonzeros   :       1
Linear constraints   :     205  [Less: 114,  Equal: 91]
  Nonzeros           :     551
  RHS nonzeros       :      38
Variables            : Min LB: 0.000000         Max UB: all infinite
Objective nonzeros   : Min   : 1.000000         Max   : 1.000000
Linear constraints   :
  Nonzeros           : Min   : 0.1000000        Max   : 2.000000
  RHS nonzeros       : Min   : 100.0000         Max   : 200.0000
```

New Features: a Checklist

- *New Parameters* on page 13
- *New Settings for Familiar Parameters* on page 13
- *New Methods in Concert Technology* on page 14
- *New Routines in the Callable Library* on page 14
- *New Commands in the Interactive Optimizer* on page 15
- *Improved and Additional Error Messages* on page 15

New Parameters

Concert Technology	Callable Library	Interactive Optimizer
FinalFactor	CPX_PARAM_FINALFACTOR	set simplex finalfactor
Purpose: control whether to factor a basis after uncrush		
RINSHeur	CPX_PARAM_RINSHEUR	set mip strategy rinsheur
Purpose: set the heuristic for relaxation induced neighborhood search		
SubMIPNodeLim	CPX_PARAM_SUBMIPNODELIM	set mip limits submipnodelim
Purpose: set a limit on the number of nodes explored in a RINS subMIP		
BarQCPEpComp	CPX_PARAM_BARQCPEPCOMP	set barrier qcpconvergetol
Purpose: set the convergence tolerance for the barrier optimizer with respect to quadratically constrained problems		

New Settings for Familiar Parameters

Parameter	New Settings	Effect
AdvInd	0 (default), 1, 2	Whether to use advanced basis or to crush starting vector supplied by user
DepInd	-1, 0, 1, 2, 3	Dependency checking during preprocessing
DiveType	3	Guided dive
DPriInd	5 CPX_DPRIIND_DEVEX	Devex pricing
MIPEmphasis	4 CPX_MIPEMPHASIS_HIDDENFEAS	Emphasize hidden feasibility
NodeAlg	0 (default) CPX_ALG_AUTOMATIC (mip strategy subalgorithm 0)	Automatic: let CPLEX choose the algorithm at the nodes in a MIP
BarGrowth	Default: 1e12	Default was previously 1e8

New Methods in Concert Technology

(They appear in Java notation here; C++ and C#.NET are similarly named.)

`IloCplex.Callback.getModel` to access the model that invoked the callback

`IloCplex.Callback.getNrows` to get the number of rows in the current active model

`IloCplex.Callback.getNcols` to get the number of columns in the current active model

`IloCplex.CutCallback.addLocal` to add a local cut with respect to a given constraint

`IloCplex.HeuristicCallback.isPrimalFeasible` to determine whether the most recently solved model is primal feasible

`IloCplex.HeuristicCallback.isDualFeasible to` determine whether the most recently solved model is dual feasible

`IloCplex.SolveCallback.isPrimalFeasible` to determine whether the most recently solved model is primal feasible

`IloCplex.SolveCallback.isDualFeasible` to determine whether the most recently solved model is dual feasible

`IloCplex.feasOpt` to find modifications that could lead to a feasible relaxation

`IloCplex.isQCP` to determine whether the current active model is quadratically constrained

New Routines in the Callable Library

`CPXreadcopyparam` to read nondefault parameter settings from a file into the Callable Library

`CPXwriteparam` to write nondefault parameter settings to a file from the Callable Library

`CPXfeasopt` to find modifications that could lead to a feasible relaxation

`CPXdelqconstrs` to delete quadratic constraints

`CPXaddqconstr` to add quadratic constraints

`CPXgetnumqconstrs` to get the number of constraints that contain a quadratic term

`CPXgetqconstrindex` to get the index of a quadratic constraint

`CPXgetqconstr` to get a quadratic constraint

`CPXgetqconstrname` to get the name of a quadratic constraint

`CPXgetqconstrslack` to get the slack of a quadratic constraint

`CPXgetmipqconstrslack` to get the slack of a quadratic constraint in a MIP model

`CPXqconstrslackfromx` to compute an array of slack values for quadratic constraints

`CPXgetxqxax` to get the value of the left-hand side (LHS) of a quadratic constraint for the current solution

---

New Commands in the Interactive Optimizer

`set advance (0, 1, or 2)`

`set barrier qcpconvergetol`

`set mip emphasis 4`

`set mip limits submipnodelim`

`set mip strategy dive 3`

`set mip strategy rinsheur`

`set mip strategy subalgorithm 0`

`set preprocessing dependency (-1, 0, 1, 2, or 3)`

`set simplex dgradient 5`

`set simplex finalfactor`

`display solution qcslacks`

Improved and Additional Error Messages

Error messages now distinguish between QP issues about a quadratic objective function and QCP issues about quadratic constraints. There are also other new error messages. For more detail about error messages, see the topic *Interpreting Error Codes* in the *Overview of the APIs* for a numerically ordered list of error codes with links to the reference manual, or see the group `optim.cplex.errorcodes` in the reference manual of the Callable Library for an alphabetic list with links to the reference manual.

3415 `CPXERR_TRE_FILE_OBJSEN`

1660 `CPXERR_PRM_DATA`

1661 `CPXERR_PRM_HEADER`

1437 `CPXERR_QCP_SENSE_FILE`

1016 `CPXERR_RESTRICTED_VERSION`

1232 `CPXERR_NO_DUAL_SOLN`

5012 `CPXERR_Q_NOT_SYMMETRIC`

1233 `CPXERR_DBL_MAX`

New Features: More Detail

Here's more detail about major new features in this release:

- *Quadratic Constraints* on page 17
- *QP Barrier Crossover* on page 19
- *New Setting for MIP Emphasis* on page 19
- *Integrality Tolerance* on page 20
- *Finding IIS after Dual Simplex* on page 21
- *Feasible Optimization* on page 21
- *Concert Technology for C#.NET Users* on page 21
- *Parallel Optimizers in Java* on page 22
- *Automatic Translation of Logical Constraints* on page 22
- *XML and Concert* on page 23
- *CSV File Format* on page 24
- *New Settings for Advanced Basis Indicator* on page 25
- *Preprocessing and Factorization (Uncrush)* on page 26
- *New Settings for Preprocessing Dependency Indicator* on page 27
- *New Devex Setting for Dual Pricing Indicator* on page 28
- *New Setting for MIP Strategy: Dive Type* on page 29
- *New Default Value for Barrier Growth Parameter* on page 29
- *Improved Documentation* on page 30

Quadratic Constraints

ILOG CPLEX now automatically recognizes and solves models that include certain types of quadratic constraints. For more about that topic, see the new chapter about solving quadratically constrained programs (QCP) in the *ILOG CPLEX User's Manual*.

Consider this conventional representation of a quadratic constraint: `x'Qx + ax ≤ r` (analogous to the familiar representation of an LP as `Ax = b`) where Q is a positive semi-definite (PSD) matrix.

With default settings, the following calls will recognize certain types of quadratic constraints and invoke the barrier optimizer to solve a continuous QCP.

- Callable Library routine `CPXbaropt`
- Concert Technology for C++ users `IloCplex::solve`
- Concert Technology for Java users `IloCplex.solve`
- Concert Technology for C#.NET users `Cplex.Solve`
- Interactive Optimizer `optimize`

It is possible to enter a problem with quadratically constrained variables from a file in LP format (that is, a formatted file with the `.lp` extension in its name). Enter the constraint in the **subject to** section of the file, using square brackets as delimiters, asterisk (`*`) for multiplication of variables, and caret (`^`) for exponentiation.

In Concert Technology, quadratic constraints can be referenced like linear constraints. For example, you can query information about quadratic constraints with the routines `getSlack` and `getSlacks` in a callback. In the C API of the Callable Library, a separate set of functions is available to support quadratic constraints.

There are new constants indicating solution quality associated with QCP. These constants are for use with the Concert Technology method `getQuality` or the Callable Library routines `CPXgetdblquality` or `CPXgetintquality`.

- `CPX_MAX_QCPRIMAL_RESIDUAL`

  `double`: To access the maximum residual `|x'Qx + ax - r|` over all the quadratic constraints in the unscaled problem.

  `int`: To access the lowest index over all the quadratic constraints where the maximum residual occurs in the unscaled problem.

- `CPX_SUM_QCPRIMAL_RESIDUAL`

  To access the sum of the residuals `|x'Qx + ax - r|` for the unscaled quadratic constraints.

- `CPX_MAX_QCSLACK_INFEAS`

`double`: To access the maximum infeasibility of the quadratic constraints, or equivalently, the maximum bound violation of the quadratic constraint slacks.

`int`: To access the lowest index of the quadratic constraints where the maximum quadratic slack infeasibility occurs.

- `CPX_SUM_QCSLACK_INFEAS`

  To access the sum of the infeasibilities of the quadratic constraints.

- `CPX_MAX_QCSLACK`

  `double`: To access the maximum absolute quadratic constraint slack value.

  `int`: To access the lowest index of the quadratic constraints where the maximum quadratic constraint slack values occcurs.

- `CPX_SUM_QCSLACK`

  To access the sum of the absolute quadratic constraint slack values.

There are new routines in the Callable Library to enable you to create, modify, and query a QCP model:

- `CPXaddqconstr` to a constraint containing a certain type of quadratic term
- `CPXgetnumqconstrs` to get the number of constraints that contain a quadratic term
- `CPXgetqconstrindex` to get the index of a quadratic constraint
- `CPXgetqconstr` to get a quadratic constraint
- `CPXgetqconstrname` to get the name of a quadratic constraint
- `CPXgetqconstrslack` to get the slack of a quadratic constraint
- `CPXgetmipqconstrslack` to get the slack of a quadratic constraint in a MIP model
- `CPXqconstrslackfromx`
- `CPXgetxqxax` to get the value of the left-hand side of a quadratic constraint for the current solution

There are new problem types associated with QCP:

- 10 `CPXPROB_QCP` (continuous quadratically constrained)
- 11 `CPXPROB_MIQCP` (mixed integer quadratically constrained)

There are new error code associated with QCP:

- 6002 `CPXERR_QCP_SENSE`
- 1031 `CPXERR_NOT_FOR_QCP`

There is a new solution type associated with QCP: `CPX_PRIMAL_SOLN` when only primal information is available.

QP Barrier Crossover

The barrier optimizer for QP models now has the ability to crossover to a simplex basis, as is already the case for LP models. This feature permits barrier to be used in the concurrent optimizer. This feature also make barrier more useful as the root solver of an MIQP when simplex is used to solve the nodes.

QP barrier crossover is controlled by the same `barcrossalg` parameter as for LP. With the default automatic setting, continuous QP behavior is *no crossover* (unlike the LP case). Consequently, you have to explicitly set it on to benefit from barrier crossover for QP. For MIQP (that is, discrete QP), the automatic setting will invoke a crossover step, so that subsequent nodes will gain the benefit of an advanced basis.

New Setting for MIP Emphasis

The MIP emphasis parameter now has an additional setting, 4 `CPX_MIPEMPHASIS_HIDDENFEAS` to indicate MIP emphasis on hidden feasible solutions. With this setting, the MIP optimizer works hard to find high quality feasible solutions that are otherwise very difficult to find. Use this setting when you more are interested in a good feasible solution than a provably optimal solution, and when feasibility emphasis has difficulty finding solutions of acceptable quality.

Table 3 recapitulates the previous settings of this parameter and adds the new setting.

Table 3 *New Setting for MIPEmphasis or CPX_PARAM_MIPEMPHASIS*

Setting	Symbolic Constant	Effect
0 (default)	`CPX_MIPEMPHASIS_BALANCED`	Balance optimality and feasibility
1	`CPX_MIPEMPHASIS_FEASIBILITY`	Emphasize feasibility over optimality
2	`CPX_MIPEMPHASIS_OPTIMALITY`	Emphasize optimality over feasibility
3	`CPX_MIPEMPHASIS_BESTBOUND`	Emphasize moving best bound
4	`CPX_MIPEMPHASIS_HIDDENFEAS`	Emphasize hidden feasible solutions

To set this parameter, use:

- `MIPEmphasis` in Concert Technology
- `CPX_PARAM_MIPEMPHASIS` in the Callable Library
- `set mip emphasis` in the Interactive Optimizer

See *Relaxation Induced Neighborhood Search (RINS)* on page 20 for additional considerations that affect this feature.

Relaxation Induced Neighborhood Search (RINS)

This version of ILOG CPLEX supports relaxation induced neighborhood search (RINS) for MIP, MIQP, and MIQCP. This feature introduces two new parameters:

- `CPX_PARAM_RINSHEUR` in the Callable Library and `RINSHeur` in Concert Technology
- `CPX_PARAM_SUBMIPNODELIM` in the Callable Library and `SubMIPNodeLim` in Concert Technology

There are also RINS commands in the Interactive Optimizer:

- `set mip strategy rinsheur`
- `set mip limits submipnodelim`

RINS is a heuristic that explores a neighborhood of the current incumbent to try to find a new, improved incumbent. It formulates the neighborhood exploration as a MIP, a subproblem known as the subMIP, and truncates the subMIP solution by limiting the number of nodes explored in the search tree.

The parameter `CPX_PARAM_RINSHEUR` controls how often RINS is invoked. A setting of 100, for example, means that RINS is invoked every hundredth node in the tree. The default is 0 (zero), a setting which means that CPLEX decides when to apply it. Any value from -1 (off) to `BIGINT` (very infrequent) is valid.

The parameter `CPX_PARAM_SUBMIPNODELIM` controls the number of nodes explored in the subMIP. Its default value of 500 is appropriate for typical models.

Integrality Tolerance

Integer tolerance for MIP models may now be set to zero. In other words, the following parameter can now be set to zero:

- `CPX_PARAM_EPINT` in the Callable Library
- `EpInt` in Concert Technology
- `set mip tolerances integrality` in the Interactive Optimizer

Other parameters controlling other tolerances, such as simplex feasibility and optimality, have **not** changed in a similar way. As one consequence, ILOG CPLEX no longer automatically resets simplex feasibility tolerance to be less than or equal to the integrality tolerance. (This new feature may have implications for users who relied on the automatic reset in previous releases.)

Previous versions of ILOG CPLEX restricted this tolerance to be not less than 1e-9. In certain models, a nonzero integrality tolerance makes it possible for discrete variables to have solution values slightly different from integrality, with potential adverse effects elsewhere in the model. This new ability to set a zero tolerance can improve the robustness

of MIP solutions, most commonly with little if any impact on the performance of the optimizer.

ILOG recommends that users try this setting in their applications. The current default value of `EpInt` remains 1e-5. However, a future version of ILOG CPLEX may adopt zero as the new default.

Finding IIS after Dual Simplex

The routine `CPXfindiis` now also works for LP models that have been determined by dual simplex to be infeasible. That is, to find irreducibly inconsistent sets, the routine no longer requires the primal simplex method, and it will not return the error code `CPXERR_IIS_NO_PRIMAL`. In consequence, the error `CPXERR_IIS_NO_PRIMAL` no longer exists.

Feasible Optimization

A new feature for exploring feasibility has been added to the ILOG CPLEX Callable Library and Concert Technology (C++, Java, and C#.NET). Its purpose is to take an infeasible model and relax bounds on variables or RHS of constraints, to show what is required to make the model feasible. For more information, see these topics in the reference manuals:

- Callable Library routine `CPXfeasopt`
- Concert Technology for C++ users `IloCplex::feasOpt`
- Concert Technology for Java users `IloCplex.feasOpt`
- Concert Technology for C#.NET users `Cplex.FeasOpt`

This feature introduces these new solution status codes:

- `CPX_STAT_FEASIBLE_RELAXED`
- `CPX_STAT_OPTIMAL_RELAXED`
- `CPXMIP_FEASIBLE_RELAXED`
- `CPXMIP_OPTIMAL_RELAXED`

The routine `CPXgetmethod` returns `CPX_ALG_FEASOPT` when this new feature is being applied to your model by ILOG CPLEX.

Concert Technology for C#.NET Users

This release of ILOG CPLEX offers a Concert Technology interface for users of C#.NET. See the online reference manual plus examples in *Getting Started* and the *User's Manual*.

Parallel Optimizers in Java

Parallel threads for the MIP, barrier, and concurrent optimizers are now supported in Concert Technology for Java users. The Concert Technology parameters `Threads`, `MIPThreads`, and `BarThreads` control whether these optimizers are parallel. (These are the same parameters that control whether an algorithm runs in parallel as for other varieties of Concert Technology.) See the *ILOG CPLEX Reference Manual* of the Java API for further details. See also the chapter about parallel threads in the *ILOG CPLEX User's Manual* for general considerations about developing applications with parallel threads.

Automatic Translation of Logical Constraints

For C++ applications, Concert Technology can automatically translate logical constraints into an equivalent formulation that the MIP optimizer of ILOG CPLEX can process. In fact, Concert can translate logical constraints built with these operators:

- `IloAnd`
- `IloOr`
- `IloIfThen`

Moreover, the following overloaded operators are also available to you for the same purpose:

- `||` (or)
- `&&` (and)
- `!` (not)

For example, to express the idea that two jobs with starting times `x1` and `x2` and with duration `d1` and `d2` must not overlap, you can write:

```
(x1 >= x2 + d2 || x2 >= x1 + d1)
```

These logical operators accept linear constraints as arguments. (In this release, they do not accept quadratic constraints.) The logical operators can be nested. For example, the following constraint can be extracted by an instance of `IloCplex`:

```
IloIfThen(env, (x >= y && x >= z), IloNot(x <= 3 || x >= 7))
```

In expressions, in addition to `IloPiecewiseLinear`, you can also use the following:

- `IloMin` over an array of numeric expressions;
- `IloMax` over an array of numeric expressions;
- `IloMin(x, cst)` over a numeric expression and a constant;
- `IloMax(x, cst)` over a numeric expression and a constant;
- `IloAbs` absolute value of a numeric expression.

For example, given an array `IloIntArray x` and a variable `IloNumVar y`, you can extract the following constraint:

```
IloMin(x) >= IloAbs(y) + z
```

Constraints using these expressions can appear in logical conditions, such as the following condition:

```
IloIfThen(env, (IloMax(x) <= 100), IloNot(x <= 3 || x >= 7))
```

XML and Concert

Concert Technology for C++ users now offers facilities to serialize a model (that is, an instance of `IloModel`) in XML. It also makes it possible to serialize a solution (an instance of `IloSolution`) in XML. See the *Concert Technology Release Notes* for more detail. The *Concert Technology C++ Reference Manual* documents the XML serialization API in the group `optim.concert.xml`. That group includes these classes:

IloXmlContext

`IloXmlContext` allows you to serialize an instance of `IloModel` or `IloSolution`. This class offers methods for reading and writing a model, a solution, or both a model and a solution together. There are examples of how to use this class in the reference manual.

IloXmlInfo

`IloXmlInfo` offers methods that enable you to validate the XML serialization of elements, such as numeric arrays, integer arrays, variables, and other extractables from your model or solution.

IloXmlReader

`IloXmlReader` creates a reader in an environment (that is, in an instance of `IloEnv`). This class offers methods to check runtime type information (RTTI), to recognize hierarchic relations between objects (that is, parents and children), and to access attributes of objects in your model or solution.

IloXmlWriter

`IloXmlWriter` creates a writer in an environment (that is, in an instance of `IloEnv`). This class offers methods to access elements and to convert their types as needed in order to serialize elements of your model or solution.

> ***Note:*** *There is a fundamental difference between writing an XML file of a model and writing an LP/MPS/SAV file of the same extracted model. If the model contains piecewise linear elements (PWL), or other nonlinear features, the XML file will represent the model as such. In contrast, the LP/MPS/SAV file will represent only the transformed model. That transformed model obscures these nonlinear features because of the automatic transformation that took place.*

CSV File Format

ILOG CPLEX now supports the file format known as CSV through XML facilities in Concert Technology for C++ users. CSV is a file format consisting of lines of comma-separated values in ordinary ASCII text. Concert Technology provides C++ classes adapted to reading data into your application from a CSV file. The constructors and methods of these classes are documented more fully in the *Concert Technology C++ Reference Manual*.

IloCsvReader

An object of this class is capable of reading data from a CSV file and passing the data to your application. There are methods in this class for recognizing the first line of the file as a header, for indicating whether or not to cache the data, for counting columns, for counting lines, for accessing lines by number or by name, for designating special characters, for indicating separators, and so forth.

IloCsvLine

An object of this class represents a line of a CSV file. The constructors and methods of this class enable you to designate special characters, such as a decimal point, separator, line ending, and so forth.

IloCsvReader::Iterator

An object of this embedded class is an iterator capable of accessing data in a CSV file line by line. This iterator is useful, for example, in programming loops of your application, such as `while`-statements.

New Settings for Advanced Basis Indicator

Formerly, the indicator of whether to use an advanced basis was a Boolean value that could only be on or off. Now you can set the advanced basis indicator according to Table 4.

Table 4 *Advanced Basis Indicator AdvInd or CPX_PARAM_ADVIND*

Setting	Effect
0 (default)	advanced basis indicator is off
1	advanced indicator is on; ILOG CPLEX uses an advanced basis supplied by the user
2	advanced indicator is on and ILOG CPLEX will crush an advanced basis or starting vector supplied by the user

If this parameter is set to 1 or 2, ILOG CPLEX uses advanced starting information when optimization is initiated. Setting 2 may be effective for solving the fixed MIP after integer optimization in cases where the percentage of discrete variables is low.

To turn on the advanced basis indicator, use:

- `AdvInd` in Concert Technology
- `CPX_PARAM_ADVIND` in the Callable Library
- `set advance` in the Interactive Optimizer

Preprocessing and Factorization (Uncrush)

A new parameter, `FinalFactor` in Concert Technology or `CPX_PARAM_FINALFACTOR` in the Callable Library, allows you to control whether ILOG CPLEX factors the basis of the full model after optimizing before terminating. Turning off this parameter may save memory for large problems.

When presolve makes changes to the model prior to optimization, a reverse operation (uncrush) occurs at termination to restore the full model with its solution. With default settings, the simplex optimizers will perform a final factorization of the basis on the full model before terminating. If you turn off the parameter `FinalFactor`, the final factorization after uncrushing will be skipped if uncrushed primal and dual solutions satisfy the feasibility and optimality conditions within the current tolerances; on large models, this can save some time and memory, but computations that require a factored basis after optimization (for example, for the computation of the condition number Kappa) may be unavailable, depending on the operations presolve performed.

Table 5 *Settings for Parameter FinalFactor or CPX_PARAM_FINALFACTOR*

Setting	Effect
1 on (IloTrue) default	CPLEX performs final factorization of basis on full model
0 off (IloFalse)	CPLEX skips final factorization

To turn off this parameter, use:

- `FinalFactor` in Concert Technology
- `CPX_PARAM_FINALFACTOR` in the Callable Library
- `set simplex finalfactor off` in the Interactive Optimizer

New Settings for Preprocessing Dependency Indicator

The ILOG CPLEX preprocessor offers a dependency checker which identifies and removes redundant constraints. Such reductions are usually most effective with the barrier optimizer, but these reductions can be applied when you are using any ILOG CPLEX optimizer. Formerly, the dependency checker was either on or off. Now, there are additional settings for this parameter that make it possible for a user to control dependency checking more precisely. Table 6 shows you the possible settings of the parameter that controls dependency checking, and indicates their effects.

Table 6 *Dependency Checking Parameter DepInd or CPX_PARAM_DEPIND*

Setting	Effect
-1 (default)	automatic: let CPLEX choose when to use dependency checking
0	turn off dependency checking
1	turn on only at the beginning of preprocessing
2	turn on only at the end of preprocessing
3	turn on at the beginning and at the end of preprocessing

To set the preprocessing dependency indicator to any value other than its default, use:

- `DepInd` in Concert Technology
- `CPX_PARAM_DEPIND` in the Callable Library
- `set pre dependency` in the Interactive Optimizer

New Devex Setting for Dual Pricing Indicator

The dual pricing indicator now allows you to indicate devex pricing. Table 7 recapitulates the previous settings for this parameter and adds the new setting 5 for devex pricing.

Table 7 *Dual Pricing Indicator DPriInd or CPX_PARAM_DPRIIND*

Setting	Symbolic Constant	Effect
0	`CPX_DPRIIND_AUTO`	Let CPLEX determine automatically
1	`CPX_DPRIIND_FULL`	Standard dual pricing
2	`CPX_DPRIIND_STEEP`	Steepest-edge pricing
3	`CPX_DPRIIND_FULL_STEEP`	Steepest-edge pricing in slack space
4	`CPX_DPRIIND_STEEPQSTART`	Steepest-edge pricing, unit initial norms
5	`CPX_DPRIIND_DEVEX`	Devex pricing

To set the dual pricing indicator, use:

- `DPriInd` in Concert Technology
- `CPX_PARAM_DPRIIND` in the Callable Library
- `set simplex dgradient` in the Interactive Optimizer

New Setting for MIP Strategy: Dive Type

The parameter that indicates what type of dives to perform now has an additional setting to allow CPLEX to perform guided dives. In guided dives, the branch direction is guided by the current incumbent, causing the MIP search to spend more time exploring potential solutions that are similar to the current incumbent, consistent with the idea that better solutions can often be found in a small neighborhood of an existing feasible solution.

Table 8 recapitulates the previous settings of this parameter and adds the new setting 3 guided dive.

Table 8 *New Setting for DiveType or CPX_PARAM_DIVETYPE*

Setting	Effect
0 (default)	Let CPLEX determine automatically
1	traditional dive
2	probing dive
3	guided dive

To set this parameter, use:

- `DiveType` in Concert Technology,
- `CPX_PARAM_DIVETYPE` in the Callable Library
- `set mip strategy dive` in the Interactive Optimizer

New Default Value for Barrier Growth Parameter

The default value for the parameter that controls the detection of an unbounded optimal face in the barrier optimizer has been increased from 1e8 to 1e12. To set this parameter to a different value, use:

- `BarGrowth` in Concert Technology
- `CPX_PARAM_BARGROWTH` in the Callable Library
- `set barrier limits growth` in the Interactive Optimizer

Improved Documentation

The reference manuals have been revised and expanded to document the components of ILOG CPLEX more conveniently. There is now a separate reference manual for users of the C Callable Library, C++ Concert Technology, Java Concert Technology, and C#.NET Concert Technology.

There is now a reference manual for the file formats supported by ILOG CPLEX, as well as a reference manual for the ILOG CPLEX Parameters. Now you can access documentation of the solution status codes alphabetically in the reference manual of the Callable Library through the group `optim.cplex.solutionstatus`, or by code number through the topic *Interpreting Solution Status Codes* in the Overview of the APIs. Likewise, you can access error messages alphabetically in the reference manual of the Callable Library through the group `optim.cplex.errorcodes` or by code number through the topic *Interpreting Error Codes* in the Overview of the APIs. The group `optim.cplex.solutionquality` offers a table of correspondence for solution quality between the symbolic constant of the Callable Library and the analogous value in an enumeration of Concert Technology.

Those reference manuals are available online now in two formats: compiled Microsoft Help (CHM) for easier integration on Microsoft platforms, and HTML for ease of use across all platforms with standard browsers. The reference manuals are also available in Portable Document Format (PDF) for easier printing.

The *ILOG CPLEX User's Manual* now includes material formerly available in the *Concert Technology User's Manual*. ILOG hopes this integration of two manuals (formerly separate) makes it easier to find explanations and examples you need more quickly.

Documentation of advanced routines of the Callable Library, formerly available only on demand from Technical Support, is now distributed with the product. See the topics *Advanced Presolve Functions* and *Advanced MIP Control Interface* in the *ILOG CPLEX User's Manual*, as well as individual routines and methods in the online reference manuals.

Index

A

advanced basis **25**
AdvInd **25**
automatically translating logical constraint **22**

B

BarQCPEpComp **13**
barrier optimizer
  crossover and **19**
  parallel support for Java users **22**
  QP and **19**
  quadratically constrained problems and **17**
BarVarUp removed **8**
BasInterval default value changed **8**

C

C#.NET **21**
comma-separated values (CSV) **24**
concurrent optimizer **19**
  parallel support for Java users **22**
conversion notes **6**
CPX_MIPEMPHASIS_HIDDENFEAS **19**
CPX_PARAM_ADVIND **25**
CPX_PARAM_BARQCPEPCOMP **13**
CPX_PARAM_BARVARUP removed **8**
CPX_PARAM_BASINTERVAL default value changed **8**
CPX_PARAM_DEPIND **27**
CPX_PARAM_DIVETYPE **29**
CPX_PARAM_DPRIIND **28**
CPX_PARAM_FINALFACTOR **13**, **26**
CPX_PARAM_MIPEMPHASIS **19**
CPX_PARAM_RINSHEUR **13**
CPX_PARAM_SIMTHREADS removed **8**
CPX_PARAM_SUBMIPNODELIM **13**
CPX_PRIMAL_SOLN **18**
CPXaddqconstr **14**
CPXdelqconstrs **14**
CPXERR_BOUNDS_BINARY error code removed **6**
CPXERR_DBL_MIN error code removed **6**
CPXERR_IIS_NO_PRIMAL error code removed **6**
CPXERR_IIS_NO_PRIMAL obsolete error code **21**
CPXERR_NOT_FOR_QCP **18**
CPXERR_QCP_SENSE **18**
CPXfeasopt **14**
CPXfindiis **21**
CPXgetmipqconstrslack **14**
CPXgetnumqconstrs **14**
CPXgetqconstr **14**
CPXgetqconstrindex **14**
CPXgetqconstrname **14**
CPXgetqconstrslack **14**
CPXgetxqxd **15**
CPXqconstrslackfromx **15**
CPXreadcopyparam **14**
CPXwriteparam **14**
crossover **19**

D

dependency checker **27**
`DepInd` **27**
devex pricing **28**
`dgradient` **28**
displaying results **11**
`DiveType` **29**
diving **29**
`DPriInd` **28**
dual pricing indicator **28**

E

`EpInt` **21**
error messages **15**

F

feasibility **21**
`FinalFactor` **13**, **26**

H

heuristic **20**

I

`IloCplex`
  Callback
    getModel **14**
    getNcols **14**
    getNrows **14**
  CutCallback
    addLocal **14**
  feasOpt **14**
  HeuristicCallback
    isDualFeasible **14**
    isPrimalFeasible **14**
  isQCP **14**
  SolveCallback
    isDualFeasible **14**
    isPrimalFeasible **14**
`IloRange` **10**
infeasible model **21**
integrality tolerance **7**
  zero and **20**
Interactive Optimizer
  new commands **15**
irreducibly inconsistent set (IIS) **21**
`isQuadratic` deprecated method **11**

M

MIP optimizer
  parallel support for Java users **22**
`MIPEmphasis` **19**

P

parallel support for Java users **22**
preprocessor **27**
probing **29**

Q

`qcpconvergetol` **13**
QP file format deprecated **8**
quadratic constraint **17**
  Callable Library routines for **18**
  detecting presence of **11**
  notation for **17**
  problem types **18**
  solution quality and **17**
quadratic objective function **11**

R

relaxation induced neighborhood search (RINS) **20**
relaxing bounds **21**
`RINSHeur` **13**

S

semi-continuous variable **7**
semi-integer variable **7**
simplex feasibility tolerance **7**
solution quality **17**
solution status **21**
`SubMIPNodeLim` **13**

T

tolerance
feasibility **7**
integrality **7**

V

variable
converting type **7**

X

XML **23**